confluent

Databases are like Onions

Commit Logs in an age of Microservices

@tlberglund

Normal Applications

(i.e., monoliths)

Monoliths are hard to think about.

Monoliths are hard to change.

Re-integration

There are no good ways to integrate microservices.

There are no-good ways to integrate microservices.

Filesystem

Database

Integrating Microservices through the database

- Easy. “I have a database and I know how to use it.”
- Problem: eventually causes services to co-mingle.
- Results in violating the “bounded context.”
- Great to use inside a service boundary!
- Terrible for sharing data or negotiating change.

RPC

Integrating microservices via RPC

- Avoids problems of database integration
- Feels natural given imperative programming sensibilities
- Aligns with the request/response paradigm
- Problem: cascading failures
- Question: how do you debug this system?
- Answer: *you build a log.* 🤔

Events?

Events.

What’s an event?

A shared narrative describing the evolution of the business over time.

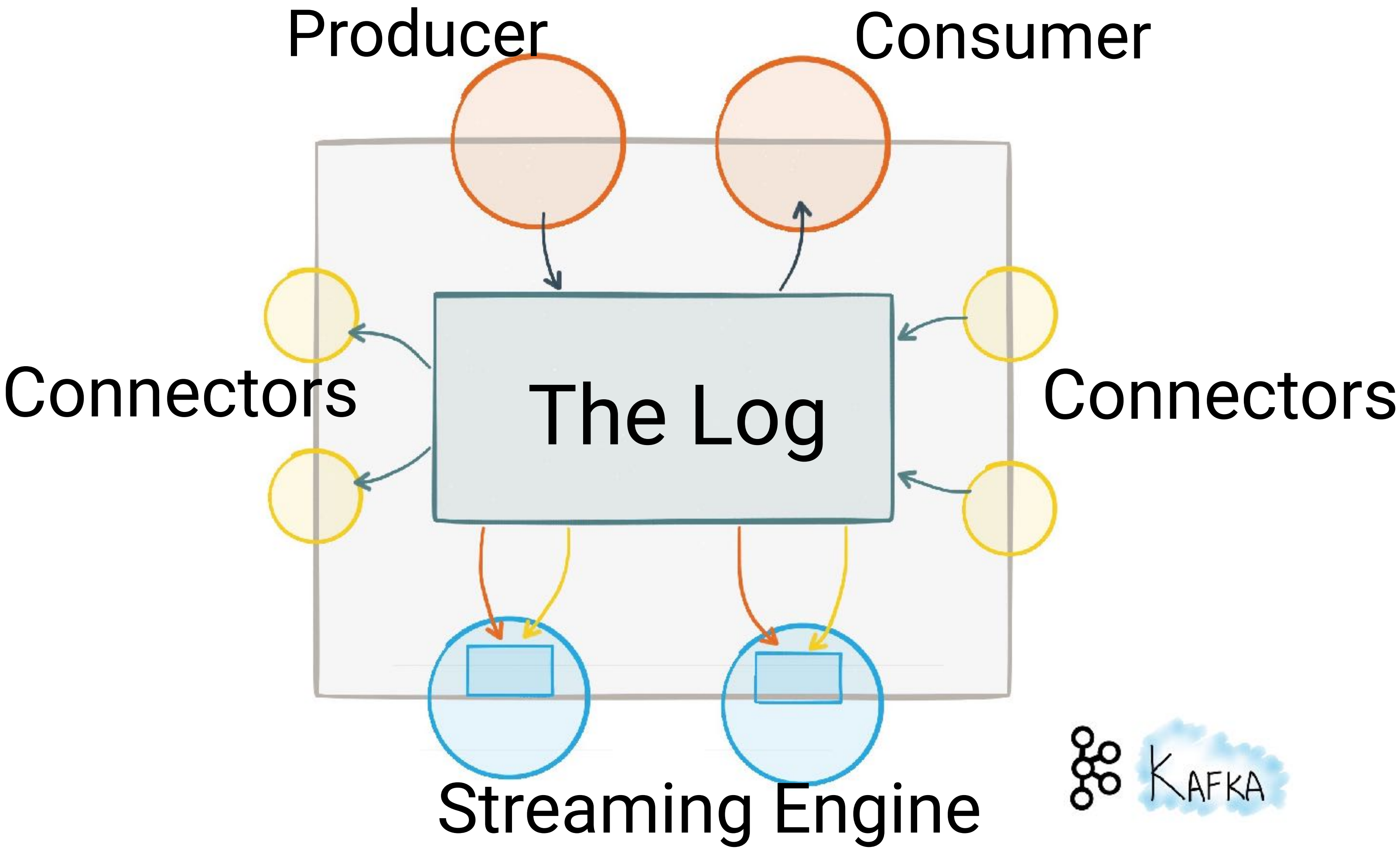
A combination of:

- Notification
- State transfer

Also, events are immutable.

Kafka Basics

What is a Streaming Platform?



Producer

Consumer

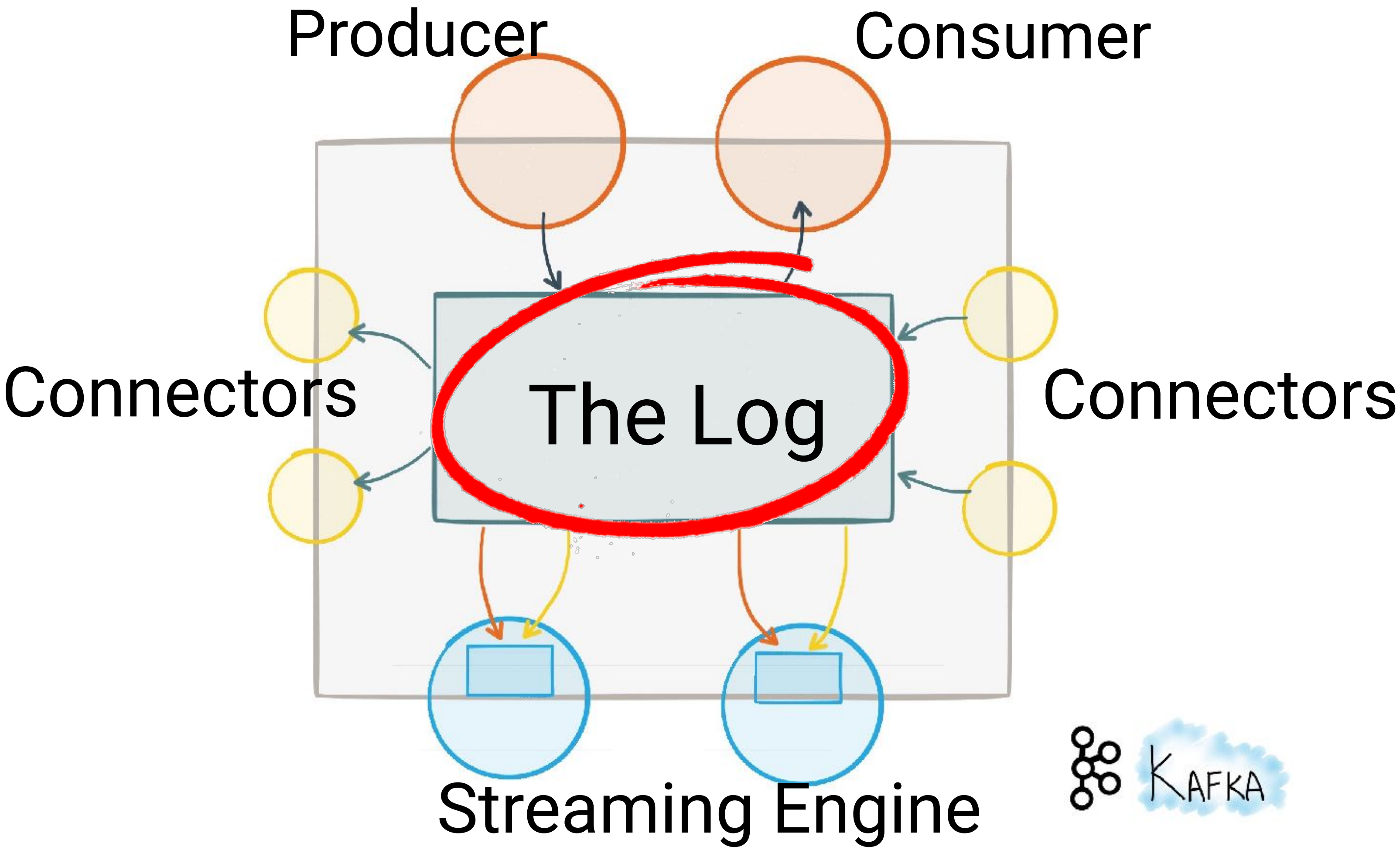
Connectors

The Log

Connectors

Streaming Engine

Kafka's Distributed Log



Producer

Consumer

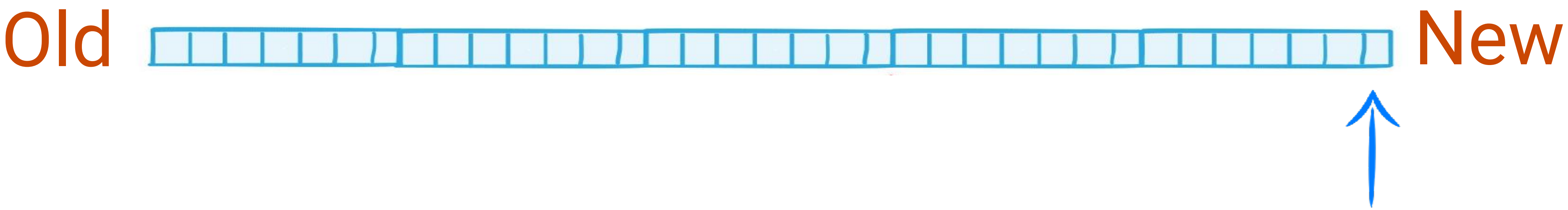
Connectors

The Log

Connectors

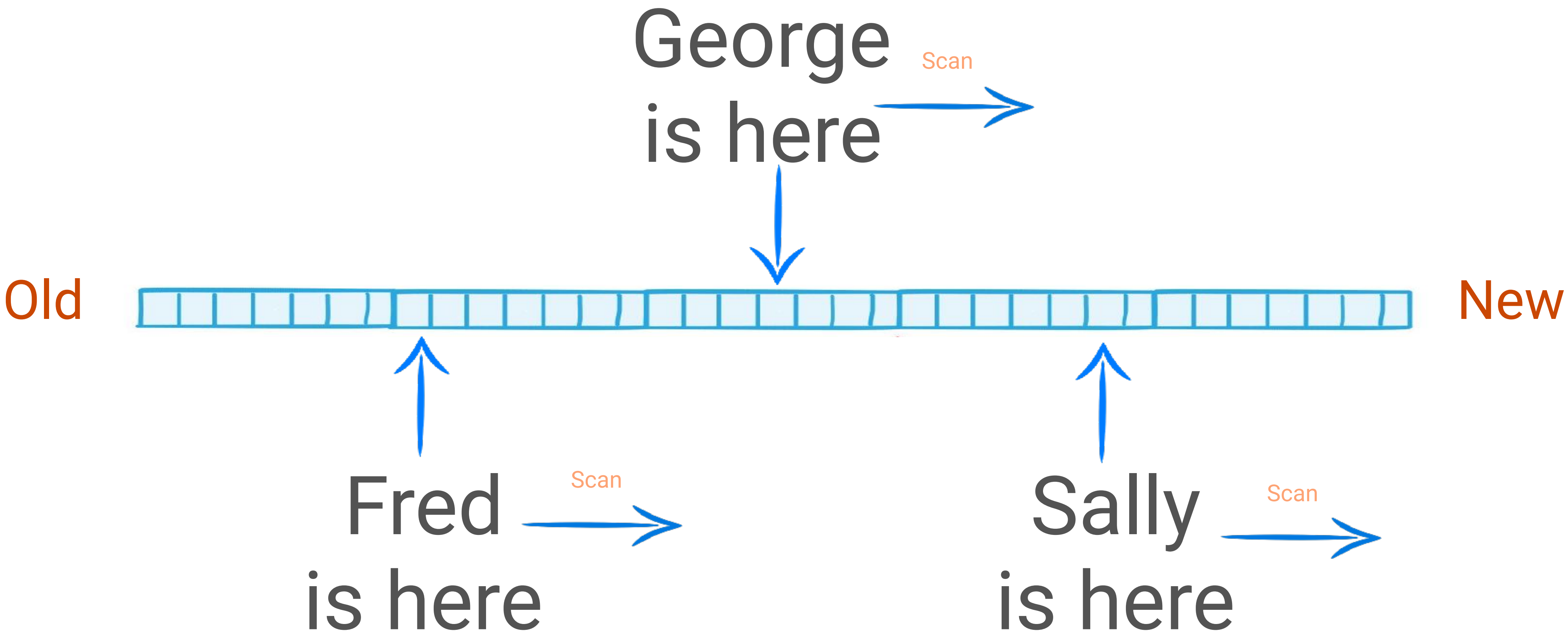
Streaming Engine

The log is a simple idea



Messages are added at the end of the log

Consumers have a position all of their own



George is here

Scan

Old

New

Fred is here

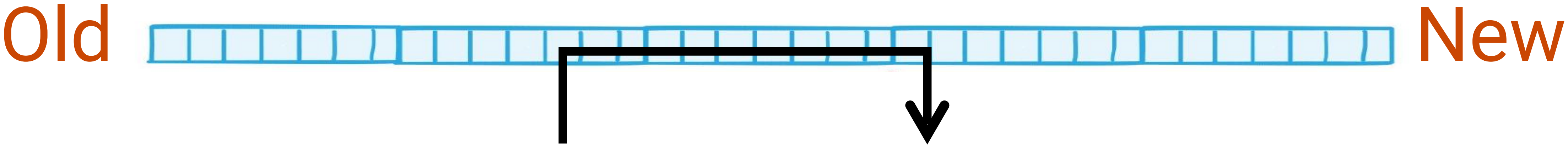
Scan

Sally is here

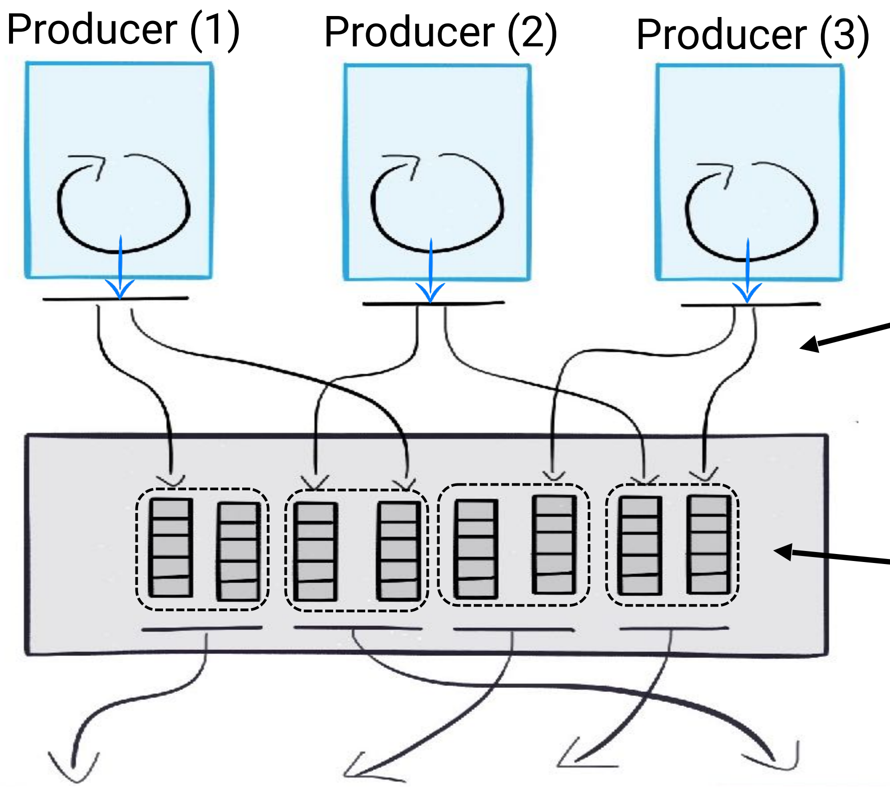
Scan

Only Sequential Access

Read to offset & scan



Shard data to get scalability

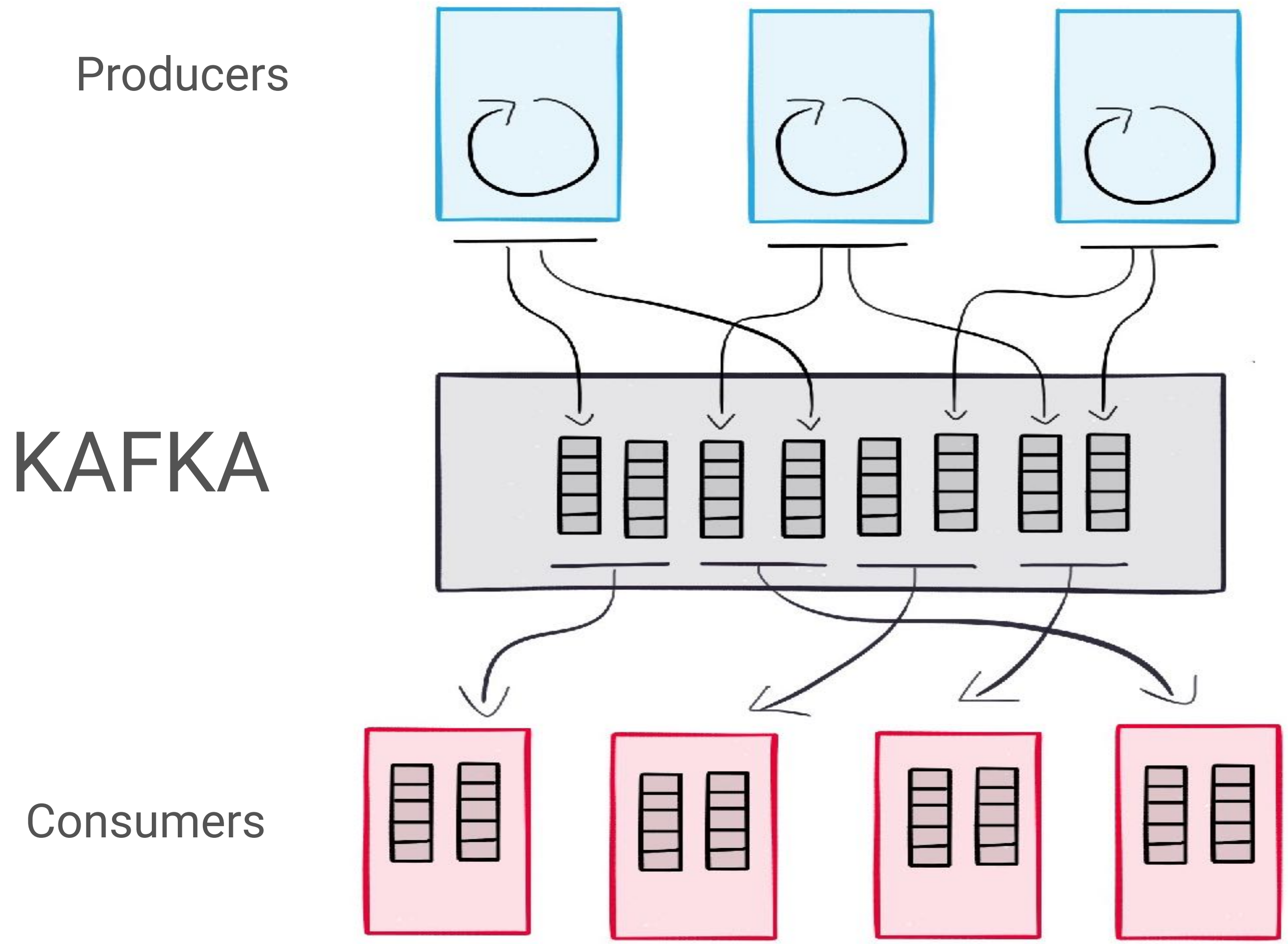


Cluster of machines

Messages are sent to different partitions

Partitions live on different machines

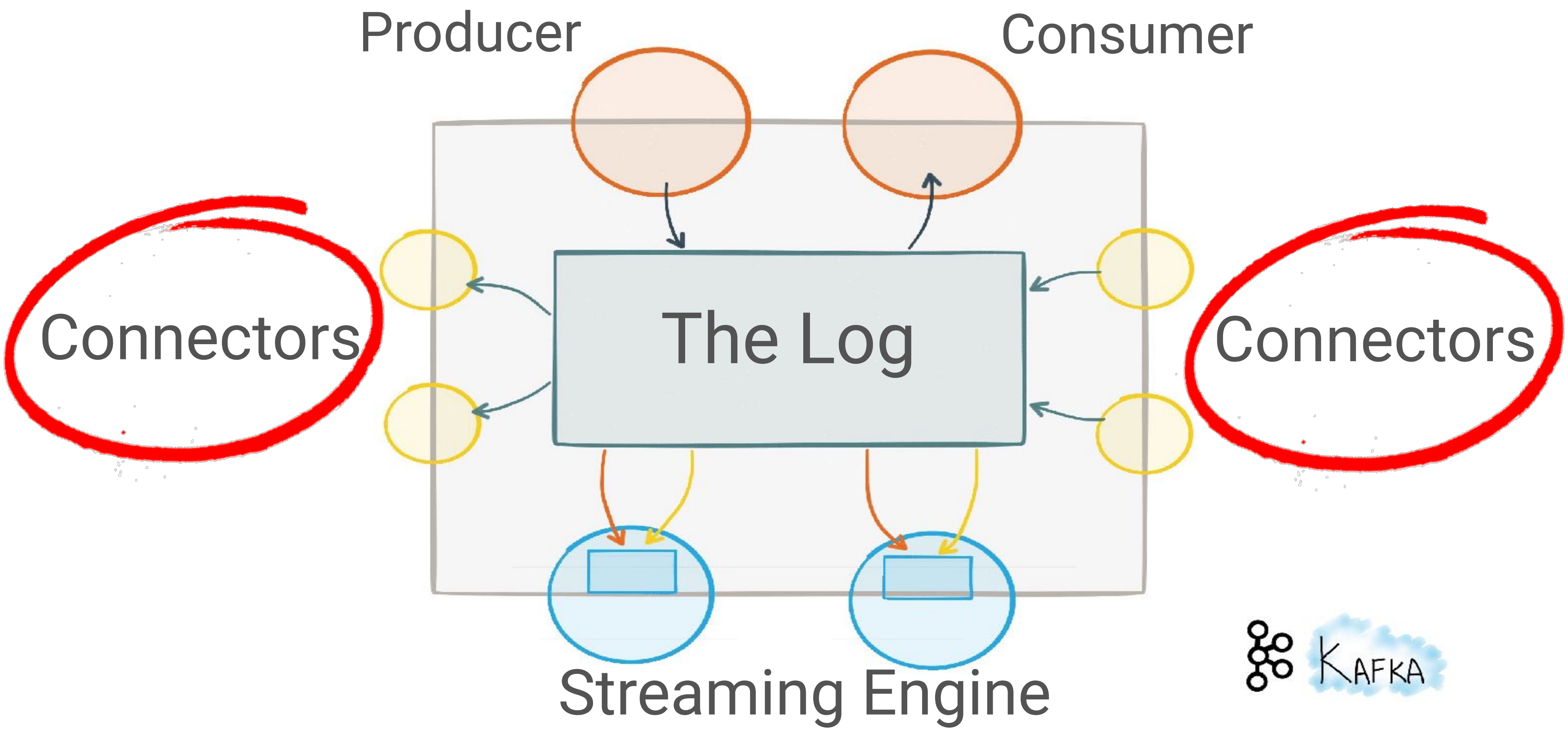
Linearly Scalable Architecture



Single topic:
- Many producers machines
- Many consumer machines
- Many Broker machines

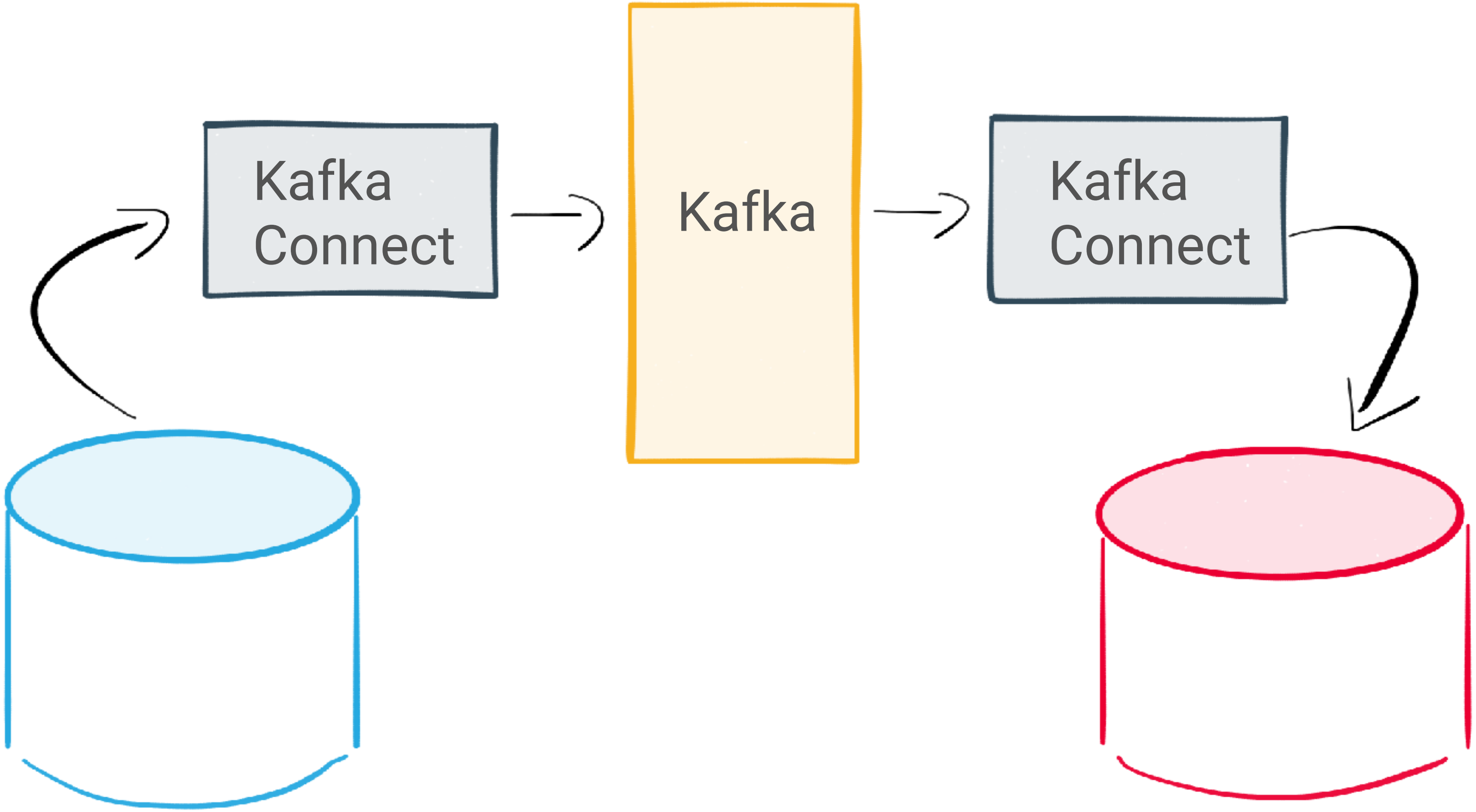
No Bottleneck!!

The Connect API



KAFKA

Ingest / Output to practically any data source

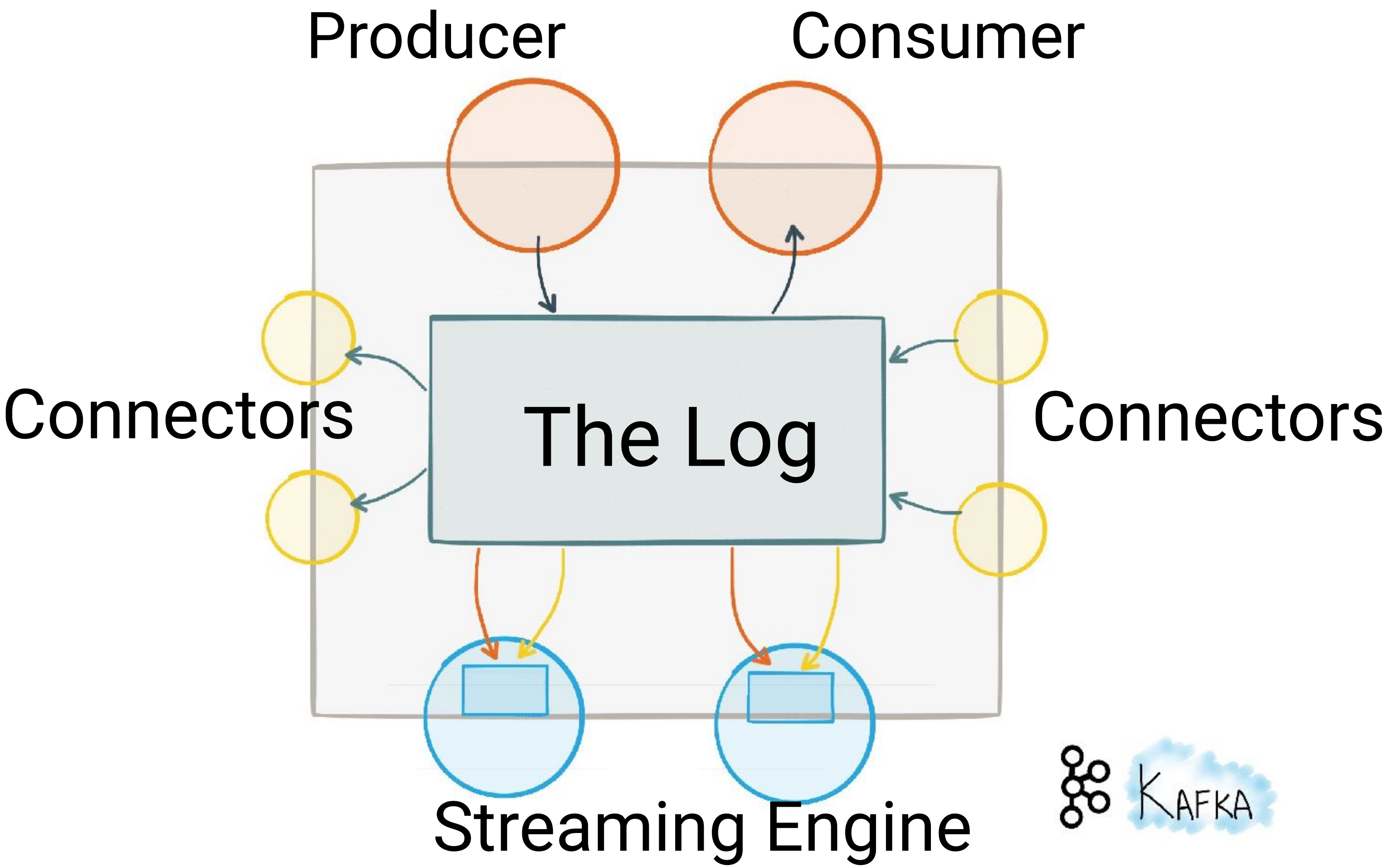


Kafka
Connect

Kafka

Kafka
Connect

Stream Processing in Kafka



Producer

Consumer

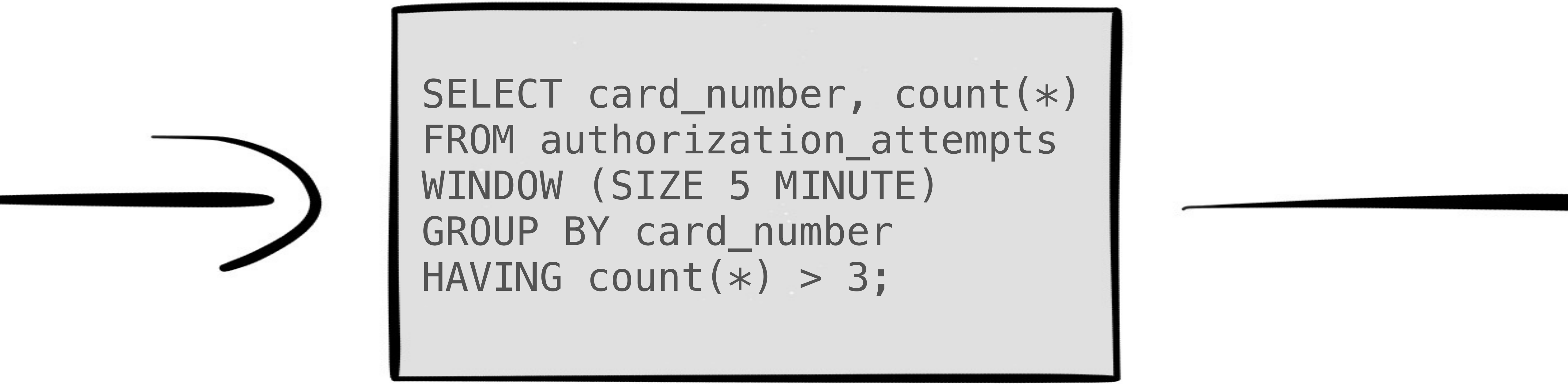
Connectors

The Log

Connectors

Streaming Engine

KSQL: an engine for continuous computation



```
SELECT card_number, count(*)
FROM authorization_attempts
WINDOW (SIZE 5 MINUTE)
GROUP BY card_number
HAVING count(*) > 3;
```

Kafka Streams: a Java API for the same

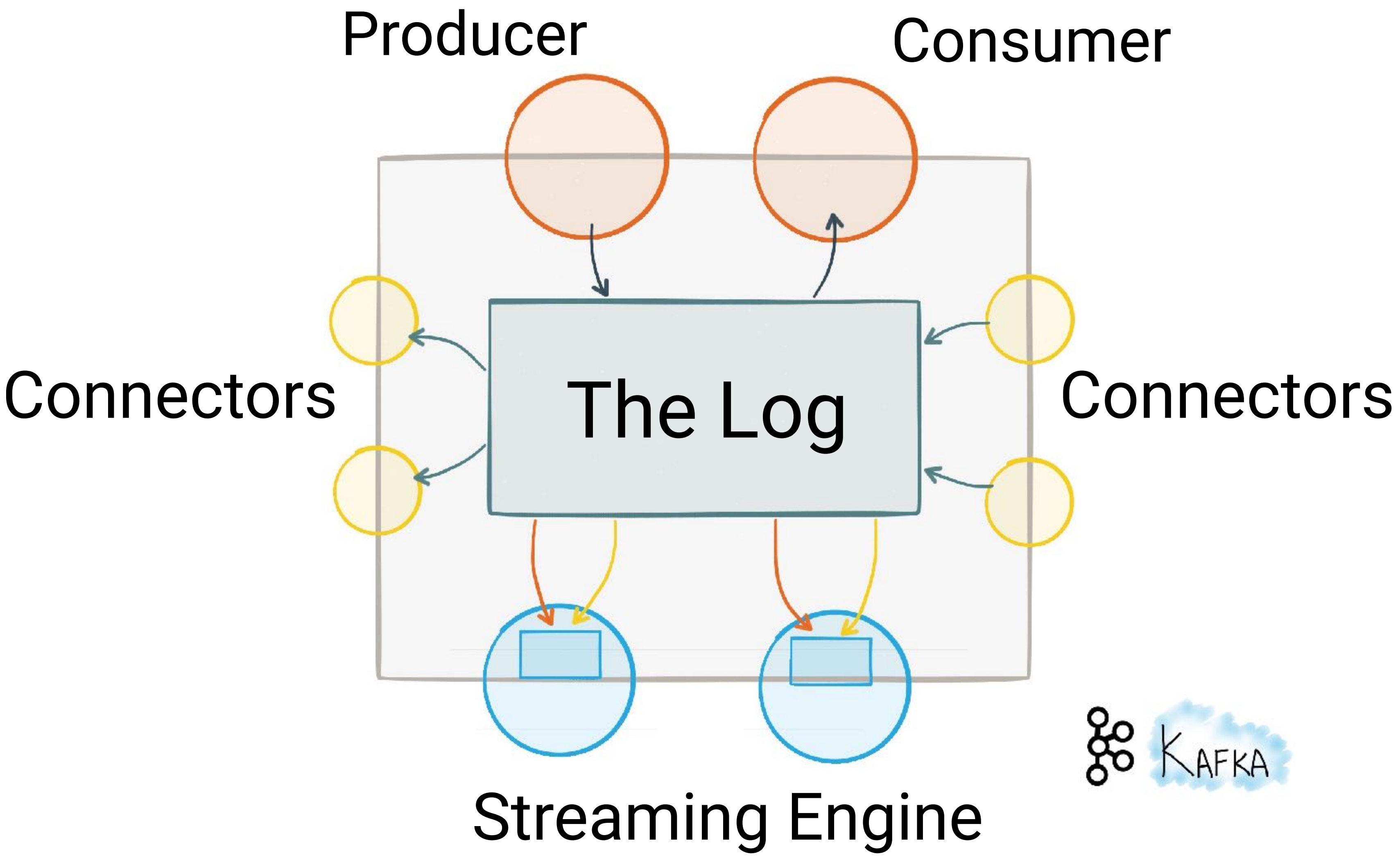
```
public static void main(String[] args) {
    StreamsBuilder builder = new StreamsBuilder();

    builder.stream(”caterpillars")
   .map((k, v) -> coolTransformation(k, v))
   .to(“butterflies”);

    new KafkaStreams(builder.build(), props()).start();
}
```

Microservices

Let's build microservices on Kafka



Producer

Consumer

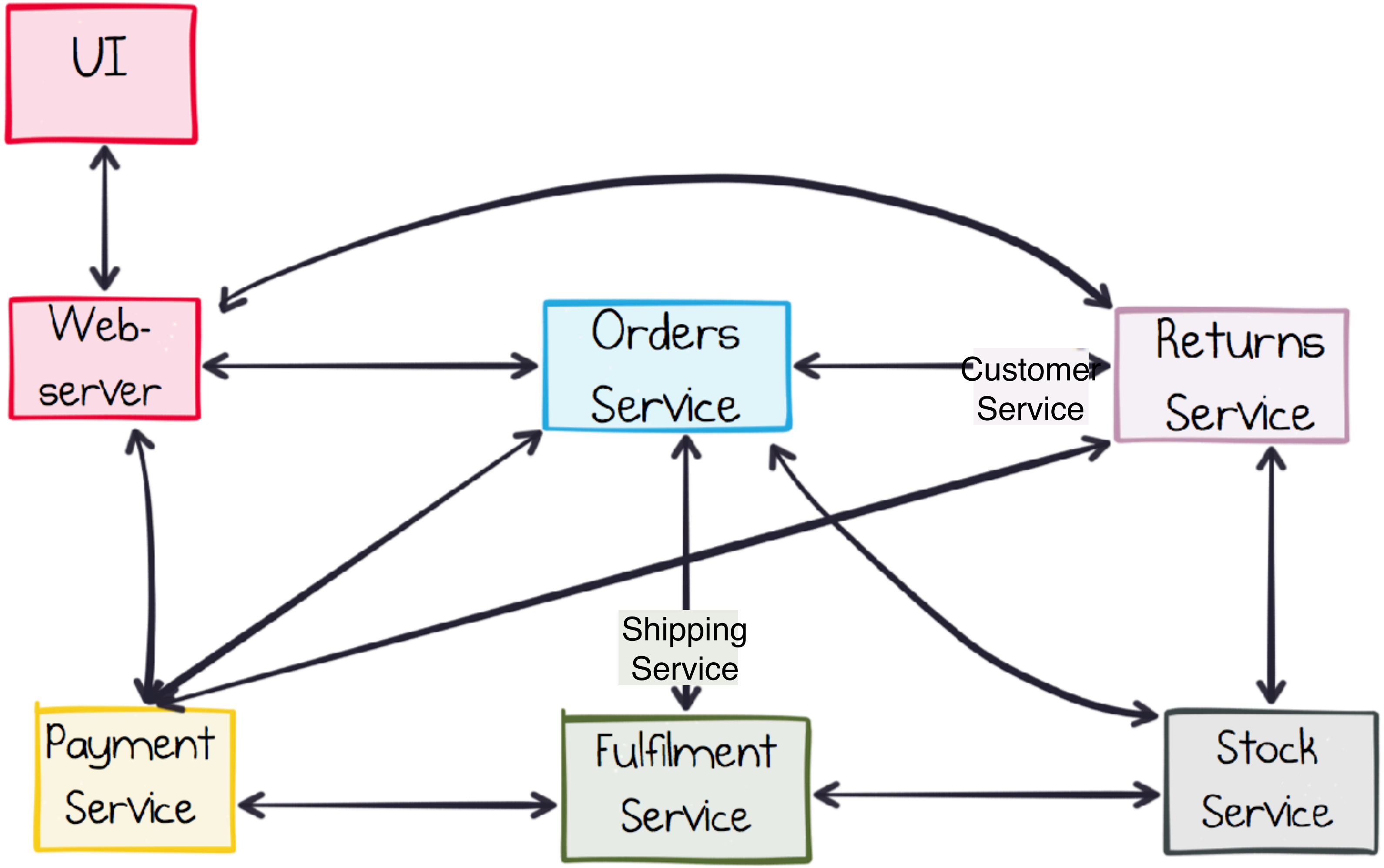
Connectors

The Log

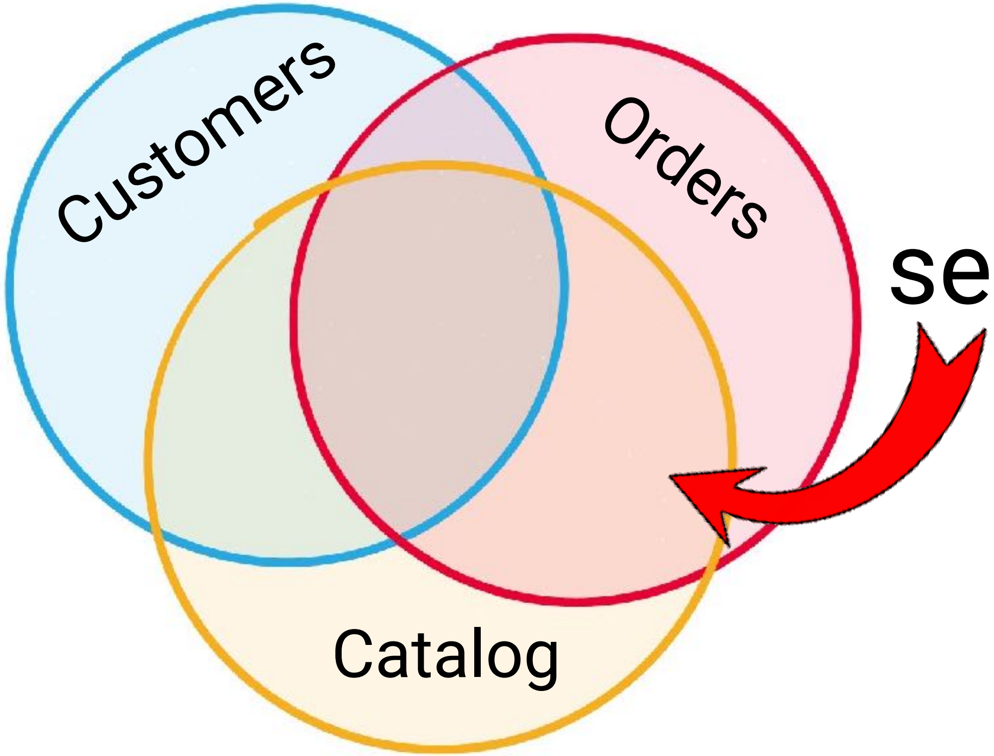
Connectors

Streaming Engine

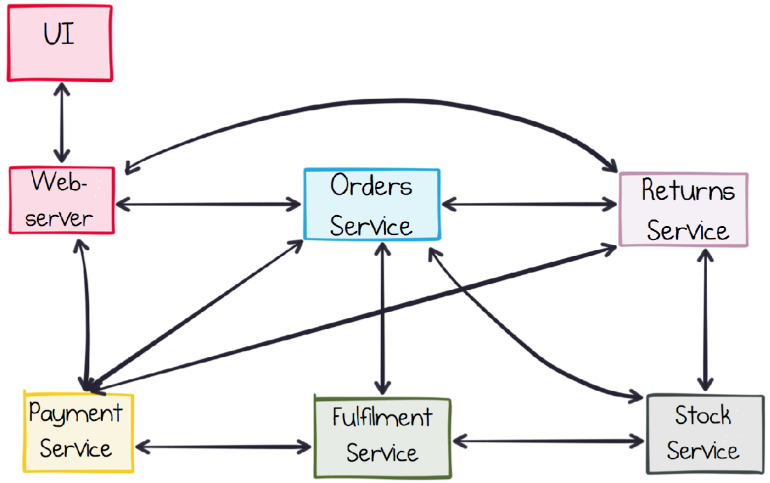
Suppose we have these services



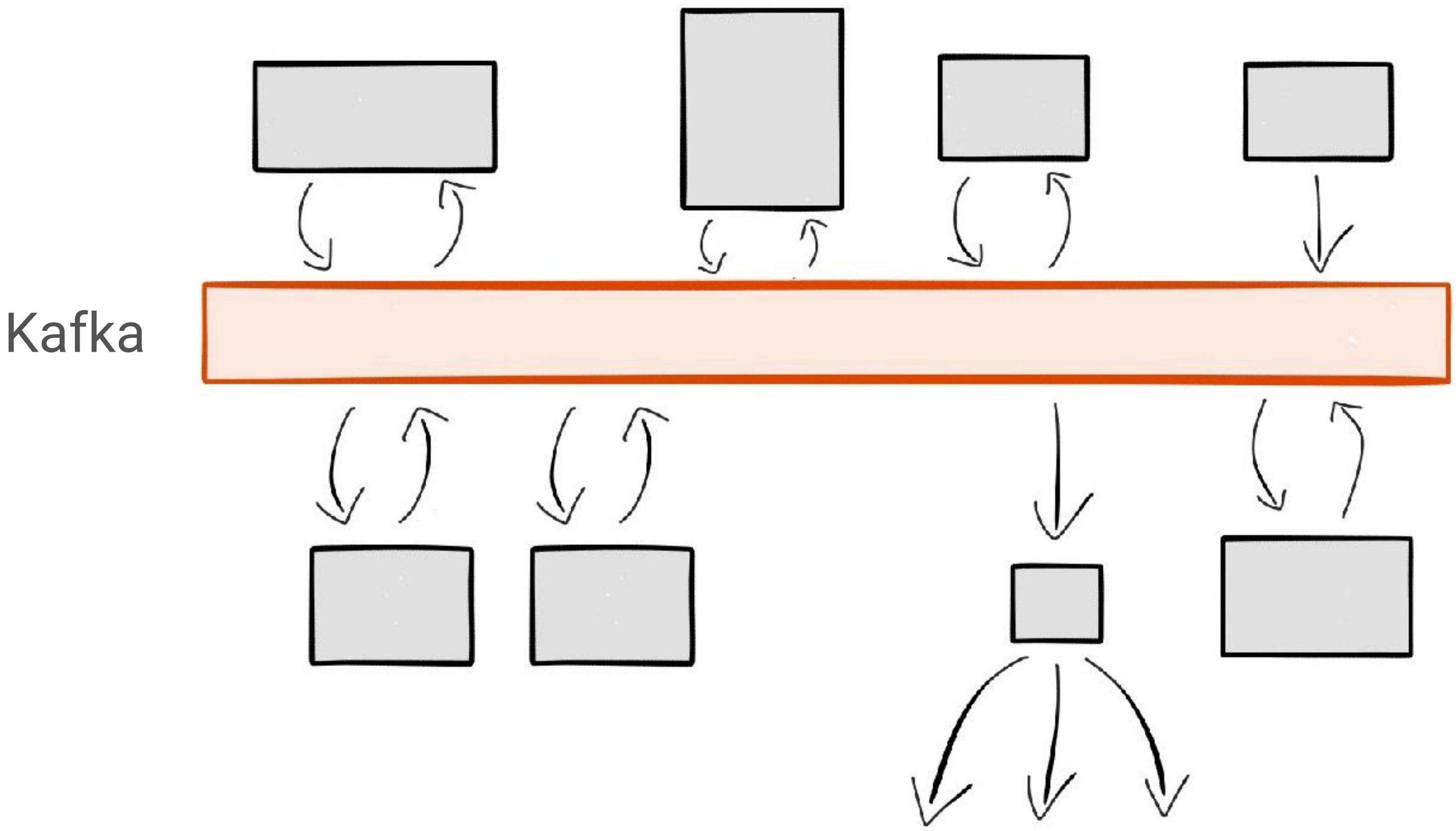
Many services share the same core facts



Most services live in here



Kafka works as a Backbone for Services to exchange Events



Notification

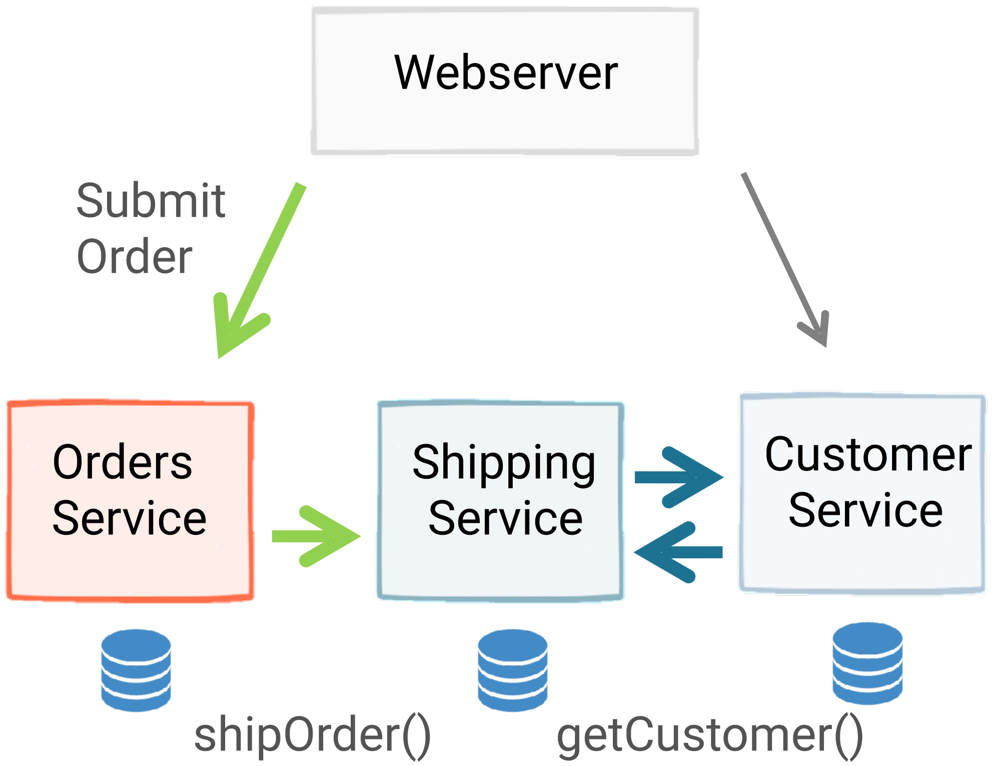
State

Recall that events wear two hats

Notification

State

ECommerce Microservices (with RPC)



Webserver

Submit Order

Orders Service

Shipping Service

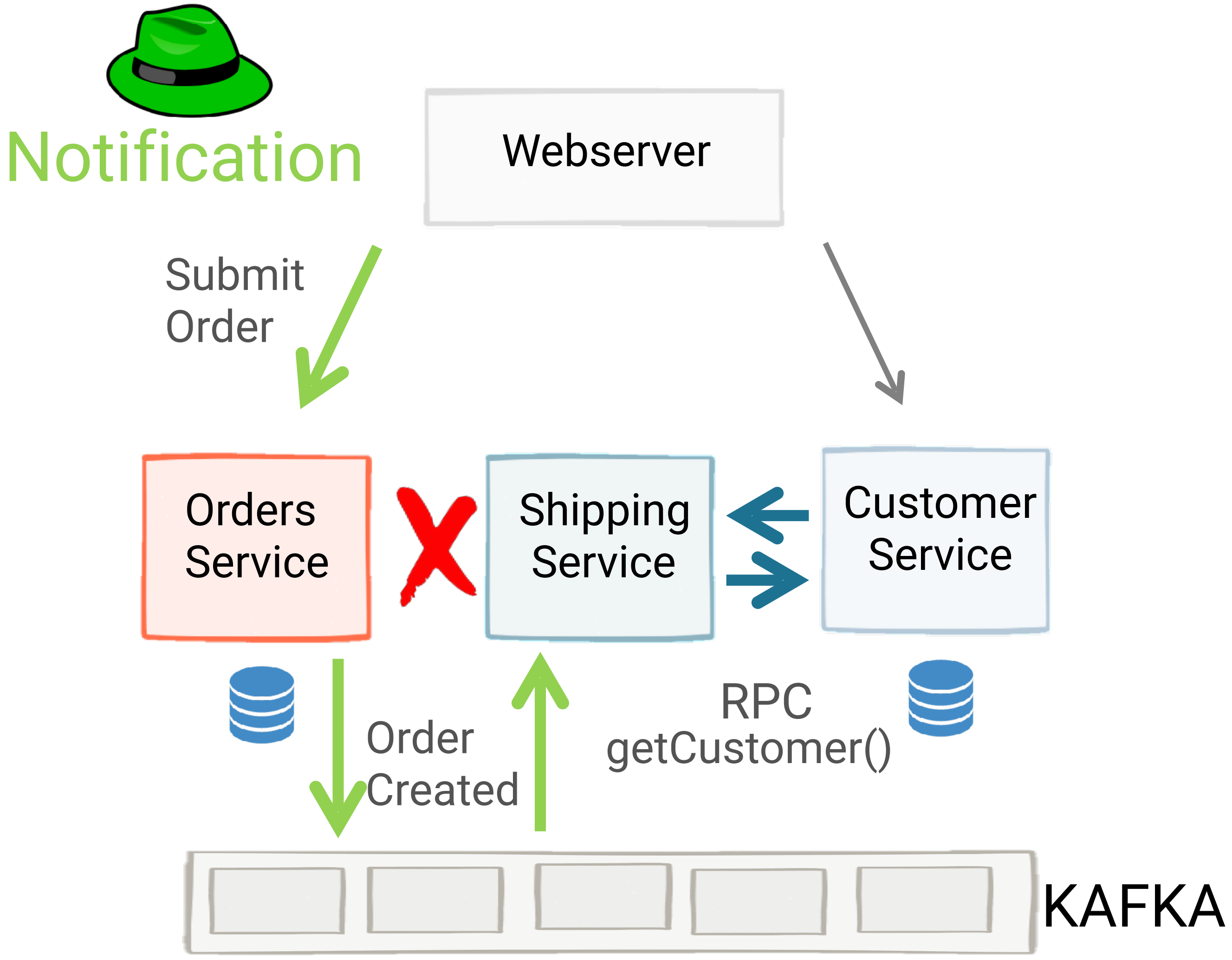
Customer Service

shipOrder()

getCustomer()

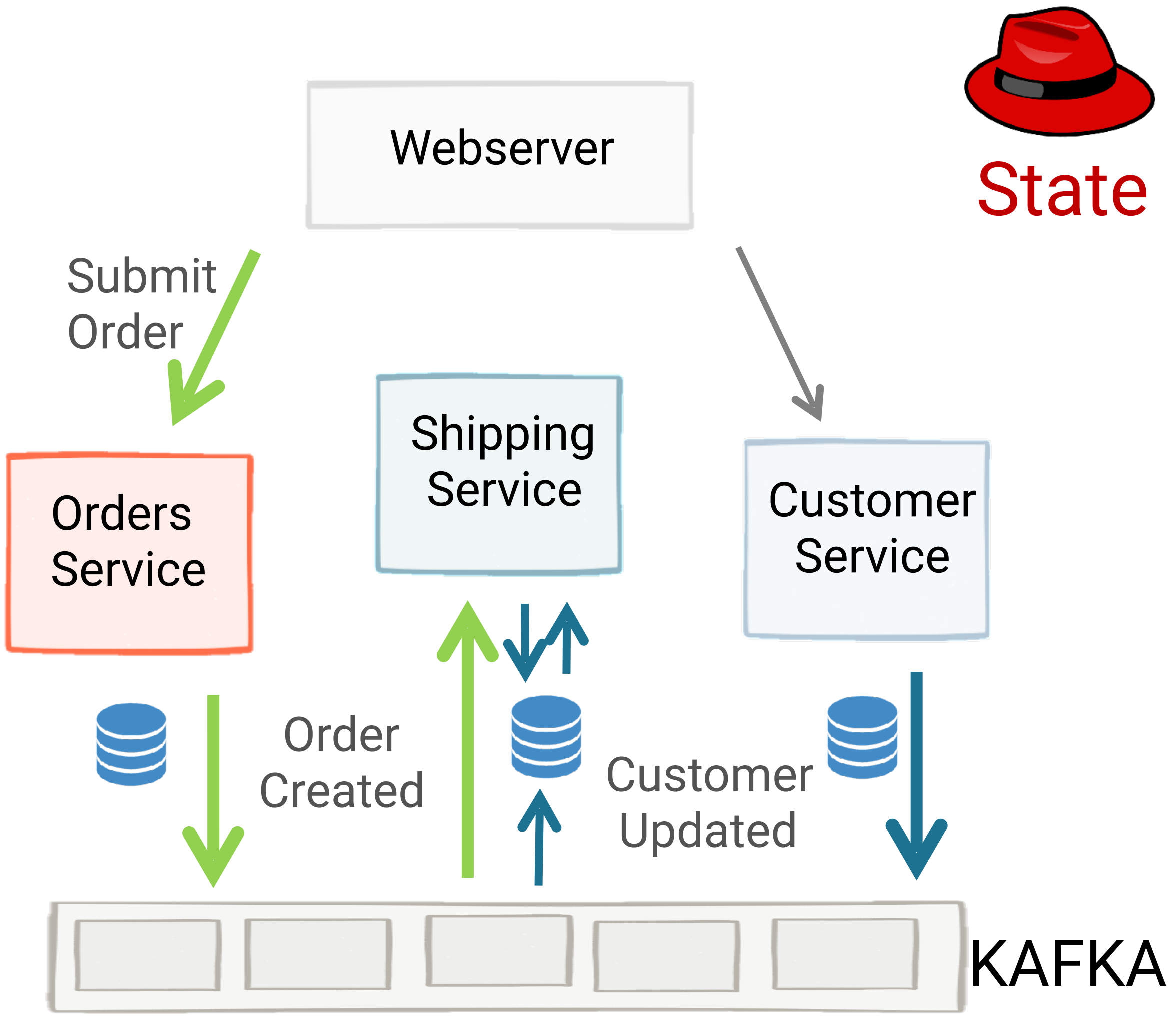
- Orders Service calls Shipping Service to tell it to ship item.
- Shipping service looks up address to ship to (from Customer Service)
- No Kafka 😢

Refactoring Orders and Shipping



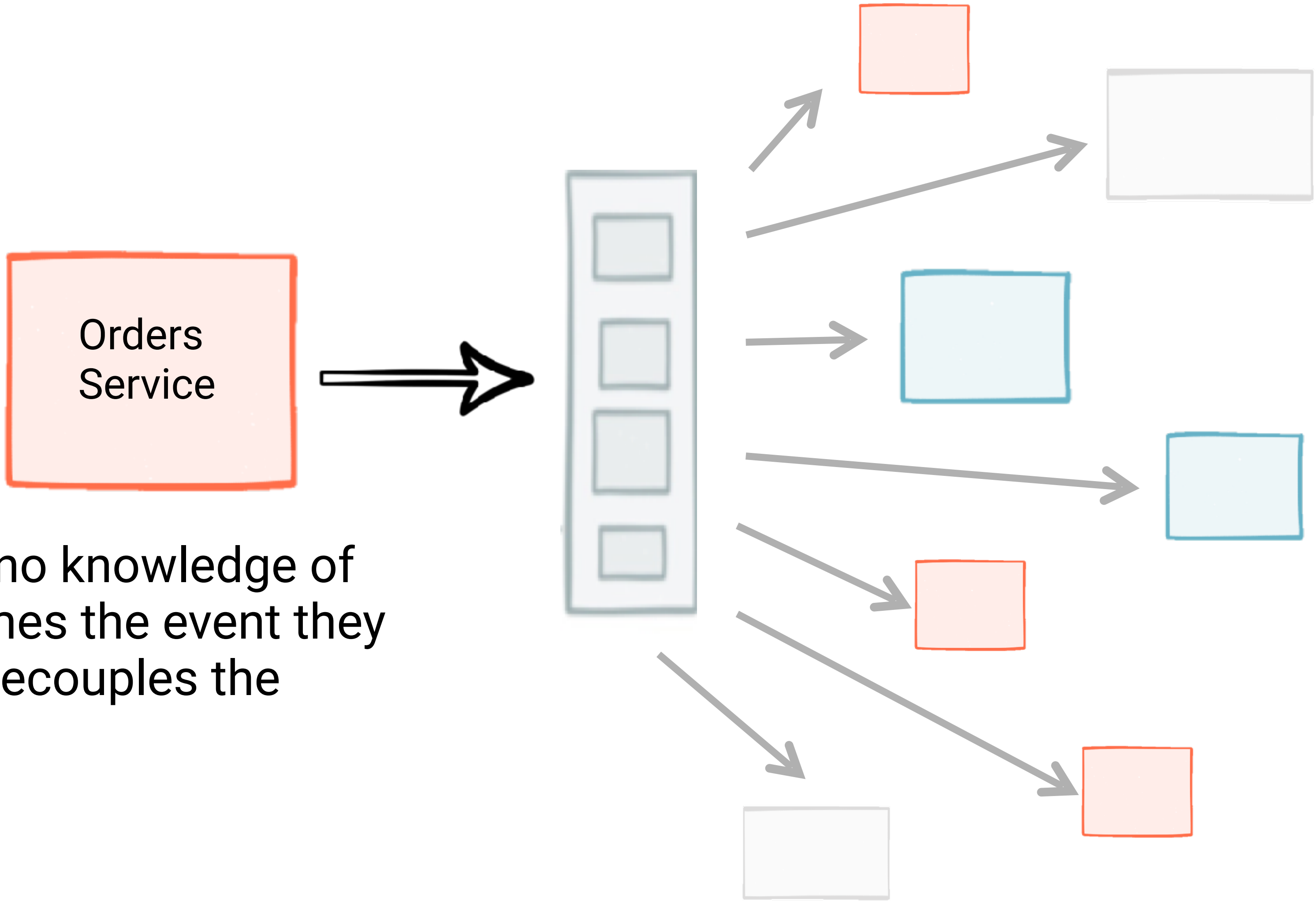
- Orders Service no longer knows about the Shipping service (or any other service). Events are fire and forget.

Refactoring Customers



- Call to Customer service is gone.
- Instead data is replicated, as events, into the shipping service, where it is queried locally.

Events are the key to scalable service ecosystems



Sender has no knowledge of who consumes the event they send. This decouples the system.

What’s a database anyway?

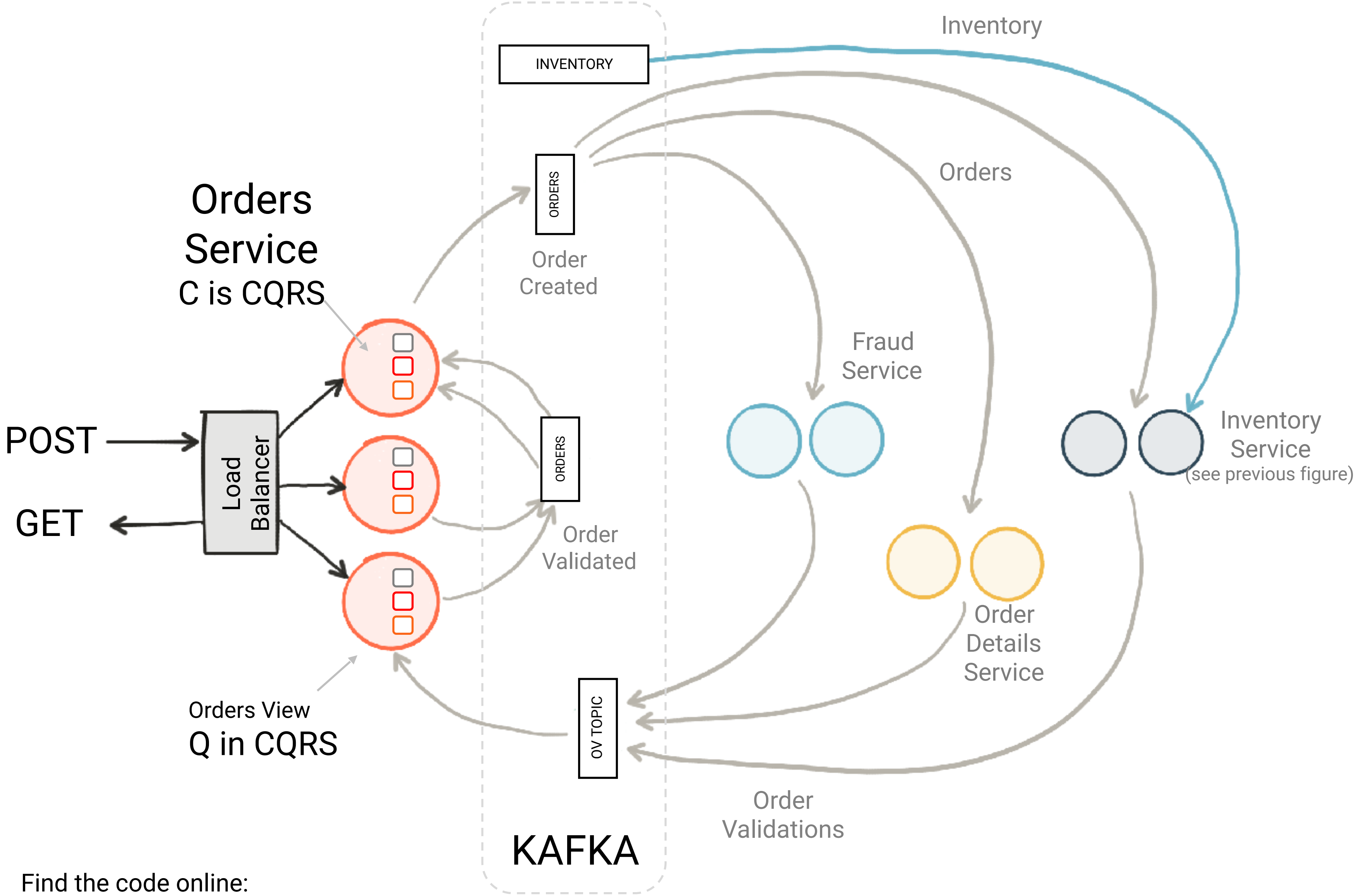
SQL

Tables

Storage Engine

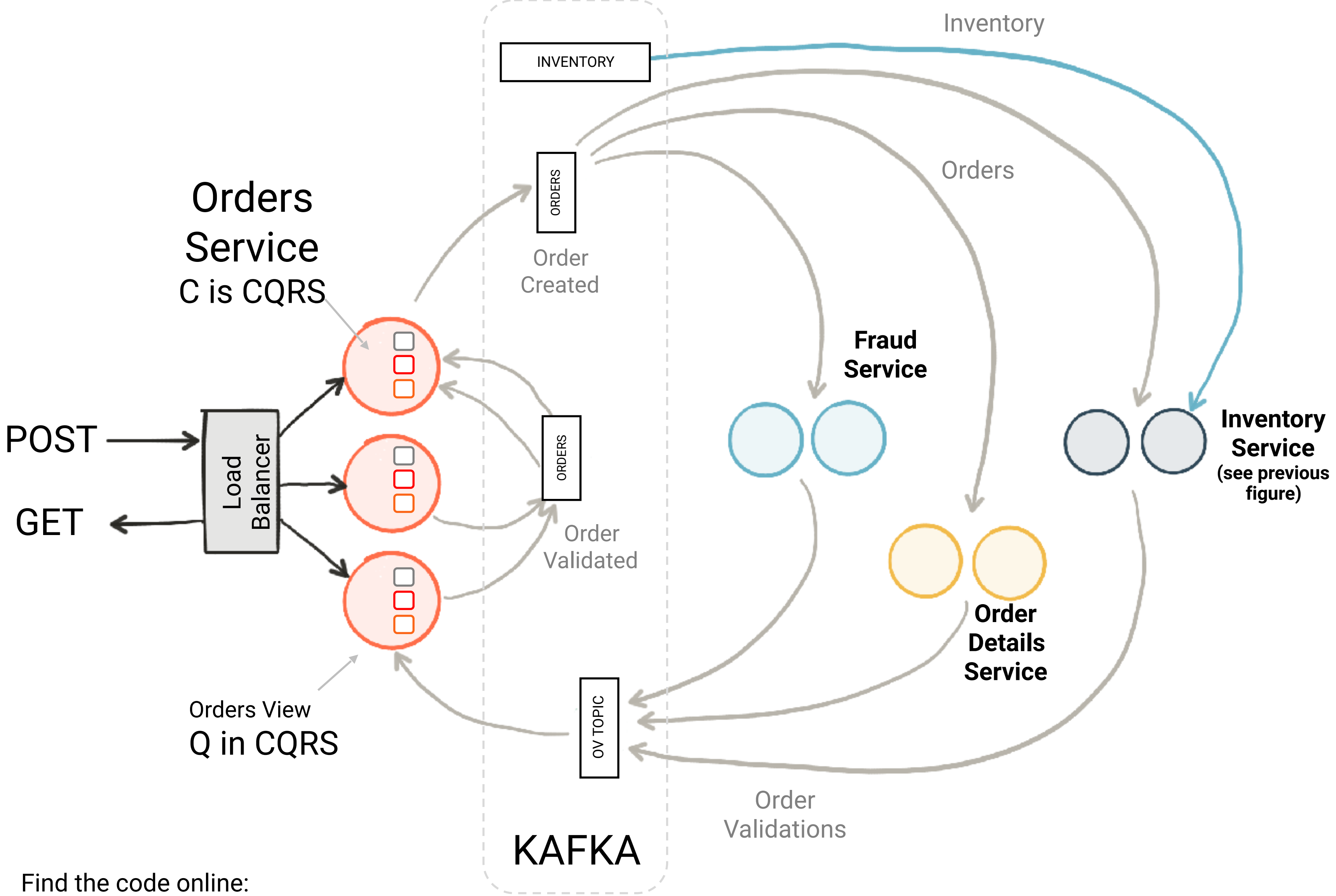
Commit Log

Consider this simple system



Find the code online:
https://github.com/confluentinc/kafka-streams-examples/tree/3.3.0-post/src/main/java/io/confluent/examples/streams/microservices

What are these things?



Find the code online:
https://github.com/confluentinc/kafka-streams-examples/tree/3.3.0-post/src/main/java/io/confluent/examples/streams/microservices

You are not just writing microservices.

You are building an inside-out database.

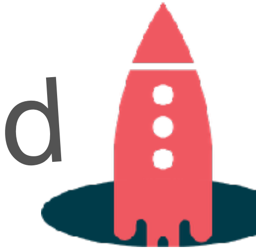
And that is a good thing.

THANK YOU

http://confluent.io/ksql

https://slackpass.io/confluentcommunity

@tlberglund



confluent